Modeling and Optimizing Hypertextual Search Engines

Based on the Reasearch of Larry Page and Sergey Brin

Michael E. Karpeles

Department of Computer Science
University of Vermont

mkarpeles@acm.org

April 15, 2009

Abstract Overview

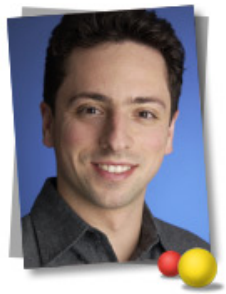
- As the volume of information available to the public increases exponentially, it is crucial that data storage, management, classification, ranking, and reporting techniques improve as well.
- The purpose of this paper is to discuss how search engines work and what modifications can potentially be made to make the engines work more quickly and accurately.
- In order to do this, we must consider the properties of the search domain (the web) difficult to mine.
- Finally, we want to ensure that our optimizations we induce will be scalable, affordable, maintainable, and reasonable to implement.

Background - Section I - Outline

- Larry Page and Sergey Brin
- Their Main Ideas
- Networking Background
- Mathematical Background

Larry Page and Sergey Brin

Larry Page was Google's founding CEO and grew the company to more than 200 employees and profitability before moving into his role as president of products in April 2001.



Brin, a native of Moscow, received a B.S. degree with honors in mathematics and CS from the University of Maryland at College Park. During his graduate program at Stanford, Sergey met Larry Page and worked on the project that became Google.

Main Idea

This presentation is based on the research paper, "The Anatomy of a Large-Scale Hypertextual Web Search Engine" written by Larry Page and Sergey Brin at Stanford University.

"The Anatomy of a Large-Scale Hypertextual Web Search Engine"

The paper by Larry Page and Sergey Brin focuses mainly on:

- Design Goals of the Google Search Engine
- The Infrastructure of Search Engines
- Crawling, Indexing, and Searching the Web
- Link Analysis and the PageRank Algorithm
- Results and Performance
- Future Work

These topics will be mentioned in this presentation.

Networking Background

- This presentation requires that you know the difference between the Web and the Internet.
- The notion of TCP and UDP connections, sockets, and ports.
- How web servers are configured to be publicly viewable.
- How CGI (The Common Gateway Interface) is used to transmit data via POST and GET

I planned on leaving myself extra time so if someone doesn't one of the areas above, I encourage you to ask away!

Mathematical Background

$$R: V \to V, \qquad V = \mathbb{R}^3$$
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \mapsto \begin{bmatrix} -y \\ x \\ z \end{bmatrix}$$
$$R \cong \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$T: V^* \times V \to \mathbb{R}$$
$$([a \quad b \quad c], \begin{bmatrix} x \\ y \\ z \end{bmatrix}) \mapsto [a \quad b \quad c] \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

The PageRank Algorithm requires previous knowledge of many key topics in Linear Algebra, such as:

- Matrix Addition and Subtraction
- Eigenvectors and Eigenvalues
- Power iterations
- Dot Products and Cross Products

I will try to avoid linear algebra when possible but some of these components are crucial to understanding PageRank.

Introduction - Section II - Outline

- Terms and Definitions
- Brief Search Engine History
- How Search Engines Work
- Search Engine Design Goals

Terms and Definitions

Hypertext

hypertext - is interactive text which is encoded with a destination address. When the hypertext link is activate (usually by a mouse click, forward arrow, or enter key) the user is redirected to the link's specified destination.

Hypertextual Search Engine

A hypertextual search engine - is a system wherein inquiries for data are parsed into sets of rules, criteria, and constraints, and then algorithmically applied to a population of data in order to ultimately isolate data candidates (search results) that best fit the original search criteria. These results are then 'reported' or displayed back to the user in hypertextual format.

Terms and Definitions, Cont'd

SERP - Search Engine Results Page

SERP - is an acronym that stands for Search Engine Result Pages. This can either relate to the physical look and feel of the results page (generated by the engine) or the process in which search engine results are categorized.

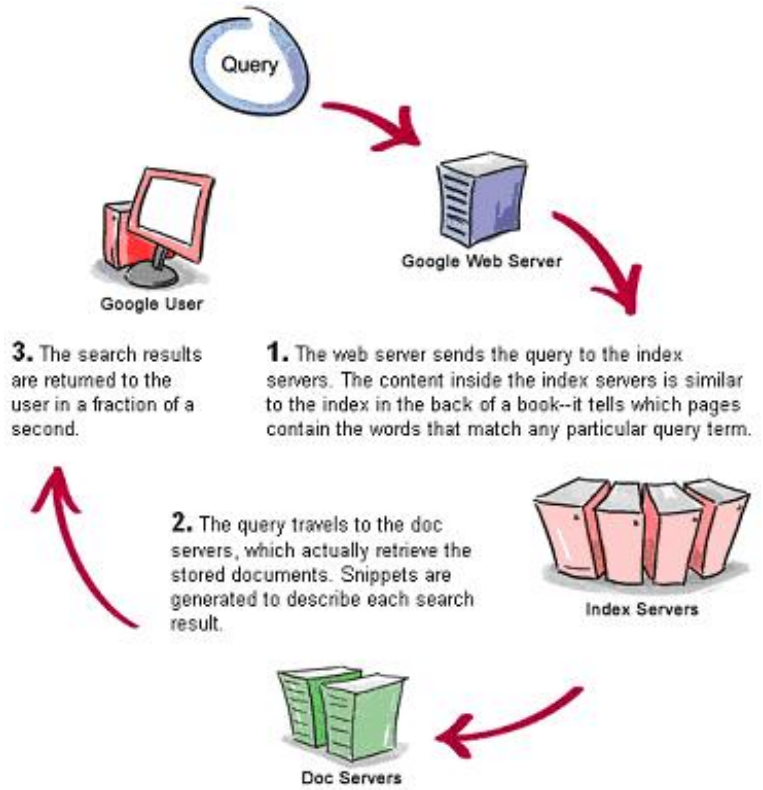
SEO - Search Engine Optimization

SEO - stands for Search Engine Optimization. SEO can either relate to search accuracy or search speed (efficiency).

Brief Search Engine History

- In 1990, Alan Emtage, Bill Heelan, and Peter Deutsch developed an engine called Archie for indexing FTP (File Transfer Protocol) archives ("The First Search Engine, Archie", Wei Li).
- While Archie was a crucial first step in creating a directory listing of FTP sites, Archie did not index the contents of these sites. Archie was followed by two search engines which followed the Gopher protocol (Port 70); Veronica (1992) and Jughead (1993).
- JumpStation (December 1993) used both a web crawling agent for searching and indexing web pages. It provided a hypertextual web form as a interface for queries. JumpStation is considered the, "first WWW resource-discovery tool to combine the three essential features of a web search engine (crawling, indexing, and searching)" (wikipedia)
- In 1994, Brian Pinkerton (University of Washington) released WebCrawler; the first search engine that provided searching of entire document content.

How Search Engines Work



- First the user inputs a query for data. he search is submitted to a back-end server via the CGI (Common Gateway Interface) Protocol. Google uses GET, as opposed to POST, for reasons that will be explained...

How Search Engines Work, Cont'd

- The server uses regex (regular expressions) to parse the user's inquiry for data. The strings submitted can be permuted, and re-arranged to test for spelling errors, and pages containing closely related content. (specifics on google's querying will be shown later)
- The search engine searches it's db for documents which closely relate to the user's input.
- In order to generate meaningful results, the search engine utilizes a variety of algorithms which work together to describe the relative importance of any specific search result.
- Finally, the engine returns results back to the user.

Google Query Evaluation

- 1. Query is parsed
- 2. Words are converted into wordIDs
- 3. Seek to the start of the doclist in the short barrel for every word.
- 4. Scan through the doclists until there is a document that matches all the search terms.
- 5. Compute the rank of that document for the query.
- 6. If we are in the short barrels and at the end of any doclist, seek to the start of the doclist in the full barrel for every word and go to step 4.
- 7. If we are not at the end of any doclist go to step 4.
- 8. Sort the documents that have matched by rank and return the top k.

Single Word Query Ranking

- Hitlist is retrieved for single word
- Each hit can be one of several types: title, anchor, URL, large font, small font, etc.
- Each hit type is assigned its own weight
- Type-weights make up vector of weights
- Number of hits of each type is counted to form count-weight vector
- Dot product of type-weight and count-weight vectors is used to compute IR score
- IR score is combined with PageRank to compute final rank

Multi-Word Query Ranking

- Similar to single-word ranking except now must analyze proximity of words in a document
- Hits occurring closer together are weighted higher than those farther apart
- Each proximity relation is classified into 1 of 10 bins ranging from a .phrase match. to .not even close.
- Each type and proximity pair has a type-prox weight
- Counts converted into count-weights
- Take dot product of count-weights and type-prox weights to computer for IR score

Search Engine Design Goals

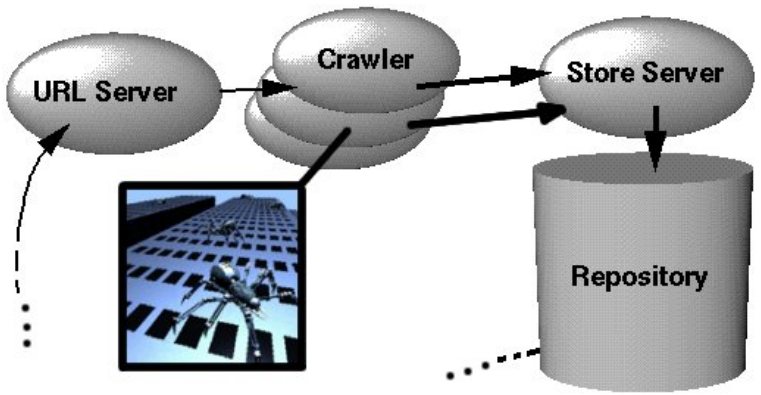
- Scalability with web growth
- Improved Search Quality
  - Decrease number of irrelevant results
  - Incorporate feedback systems to account for user approval
  - Too many pages for people to view – some heuristic must be used to rank sites' importance for the users.
- Improved Search Speed
  - Even as the domain space rapidly increases
  - Take into consideration the types of documents hosted

Search Engine Infrastructure - Section III - Outline

Google Outline

- Resolving and Web Crawling
- Indexing and Searching
- Google's Infrastructural Model

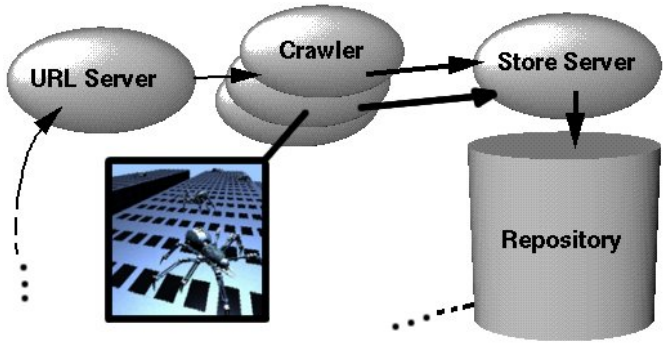
URL Resolving and Web Crawling



Before a search engine can respond to user inquiries, it must first generate a database of URLs (or Uniform Resource Locators) which describe where web servers (and their files) are located. URLs or web addresses are pieces of data that specify the location of a file and the service that can be used to access it.

The URL Server's job is to keep track of URL's that have and need to be crawled. In order to obtain a current mapping of web servers and their file trees, google's URL Server routinely invokes a series of web crawling agent called Googlebots. Web users can also manually request for their URL's to be added to Google's URLServer.

URL Resolving and Web Crawling



Web Crawlers: When a web page is 'crawled' it has been effectively downloaded. Googlebots are Google's web crawling agents/scripts (written in python) which spawn hundreds of connections (approximately 300 parallel connections at once) to different well connected servers in order to, "build a searchable index for Google's search engine" (wikipedia).

Brin and Page commented that DNS (Domain NameSpace) lookups were an expensive. Gave crawling agents DNS caching abilities.

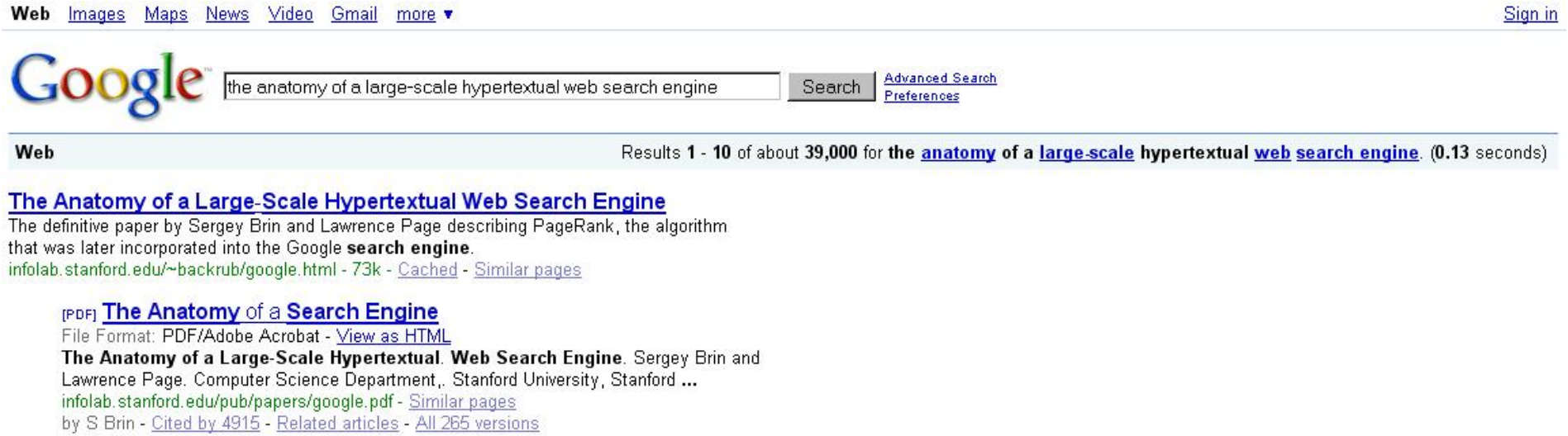
Googlebot is known as a well-behaved spider: sites avoid be crawled by adding $< metaname = "Googlebot" content = "nofollow" / >$ to the *head* of the doc (or by adding a robots.txt file)

Indexing

Indexing the Web involves three main things:

- Parsing – parsers must handle typos in meta-tags and missing data in trasnactions. Instead of using YACC to generate a CFG (Context Free Grammar) parser, Brin and Page used flex to perform lexical analysis.
- Indexing Documents into Barrels – After each document is parsed, every word is assigned a wordID. These words and wordID pairs are used to construct an in-memory hash table (the lexicon).
- Barrelling Bottleneck – Indexing Phase is difficult to make parallelized because the lexicon needs to be shared. They solved this by writing a log of 14 million words that were not in their lexicon (removed need for shared lexicon). This allowed multiple indexers to run in parallel. The log file of 14 million words is then indexed seperately.
- Sorting – the sorter takes each of the forward barrels and sorts it by wordID to produce an inverted barrel for title and anchor hits and a full text inverted barrel. This process happens one barrel at a time, thus requiring little temporary storage.

Searching

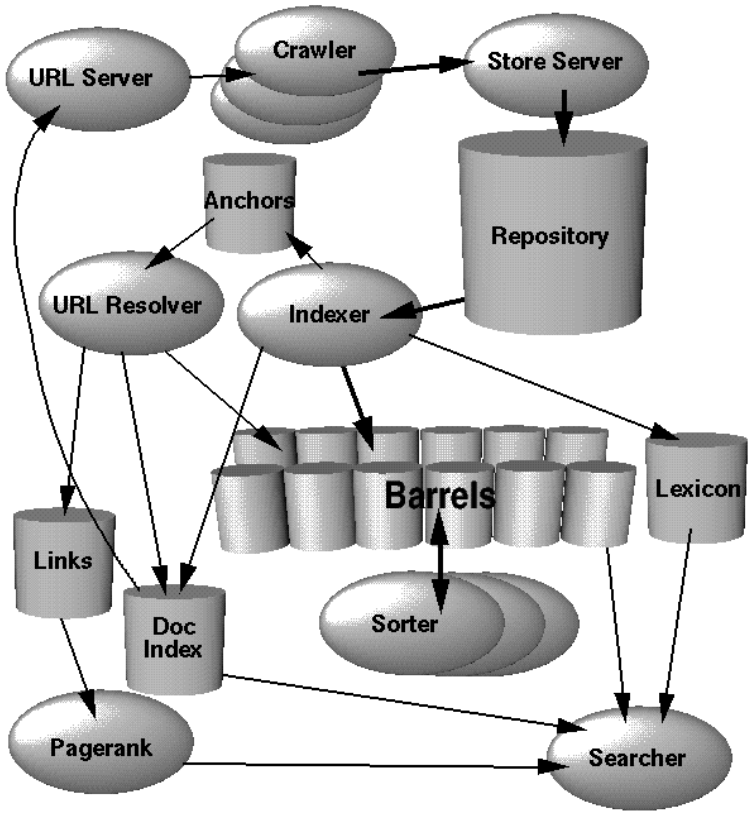


The article didn't specify any speed efficiency issues with searching. Instead they focused on making searches more accurate. During the time the paper was

written, Google queries returned 40,000 results. I only got approximately 39,000 results when I Google-seached their paper... But the results fetch only took 0.13 seconds. (Not bad!)

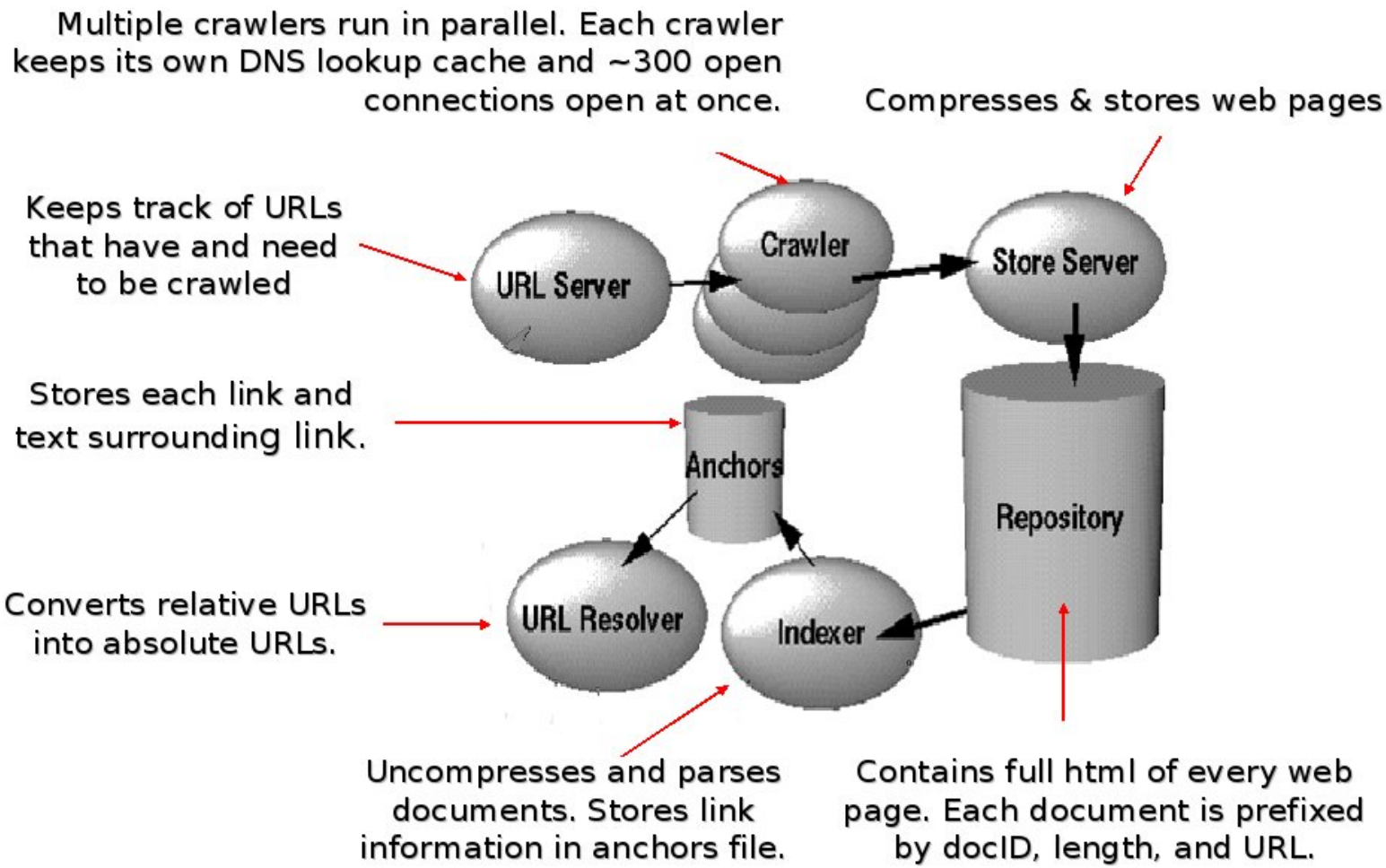
Google's Infrastructure Overview

Google's architecture includes 14 major components: an URL Server, multiple Web Crawlers, a Store Server, a Hypertextual Document Repository, an Anchors database, an URL Resolver, a Hypertextual Document Indexer, a Lexicon, multiple short and long Barrels, a Sorter Service, a Searcher Service, and a PageRank Service. These systems were implemented in C and C++ on Linux and Solaris systems.



Google's Architecture

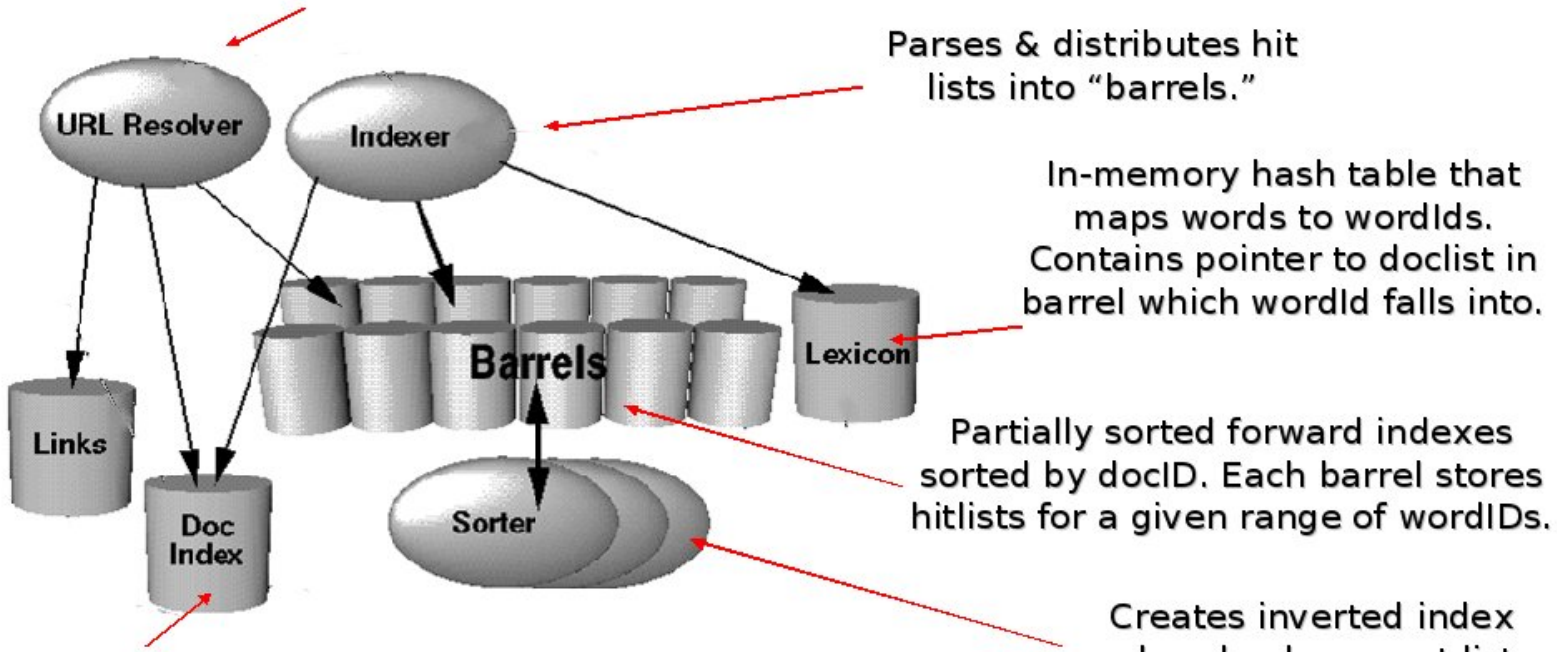
(from http://www.ics.uci.edu/~scott/google.htm)



Infrastructure Part II

Google's Architecture

(from http://www.ics.uci.edu/~scott/google.htm)

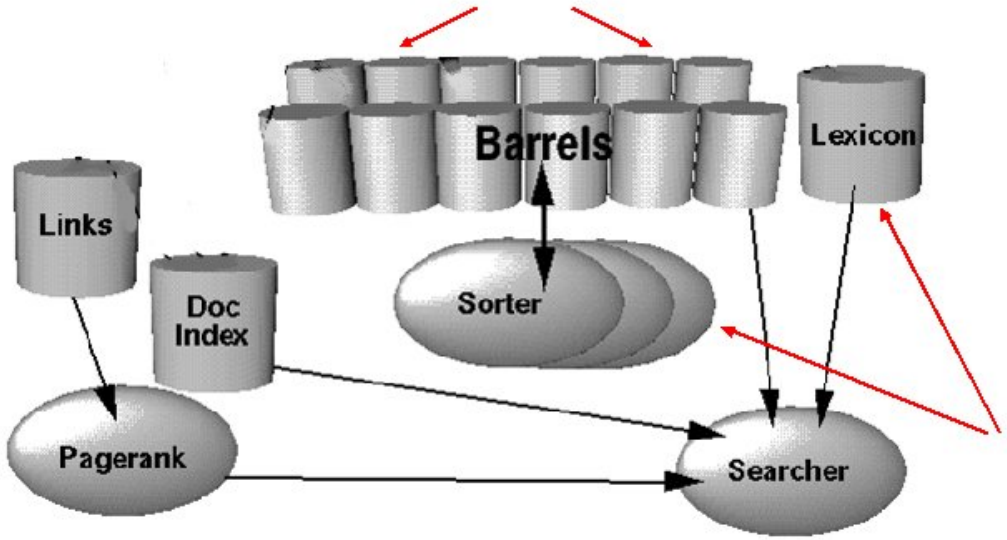


Infrastructure Part III

Google's Architecture

(from http://www.ics.uci.edu/~scott/google.htm)

2 kinds of barrels. Short barrel which contain hit list which include title or anchor hits. Long barrels for all hit lists.



List of word IDs produced by Sorter and lexicon created by Indexer used to create new lexicon used by searcher. Lexicon stores ~14 million words.

New lexicon keyed by wordID, inverted doc index keyed by docID, and PageRanks used to answer queries

Search Engine Optimizations - Section IV - Outline

- Significance of SEO's
- Elementary Ranking Schemes
- What Makes Ranking Optimization Hard?

The Significance of SEO's

- Too many sites for humans to maintain ranking
- Humans are biased – have different ideas of what "good" and "bad" are.
- With a search space as a large as the web, optimizing order of operations and data structures have huge consequences.
- Concise and well developed heuristics lead to more accurate and quicker results
- Different methods and algorithms can be combined (see: RANK MERGING) to increase overall efficiency.

Elementary SEO's for Ranking

- Word Frequency Analysis within Pages
- Implicit Rating Systems - The search engine considers how many times a page has been visited or how long a user has remained on a site.
- Explicit Rating Systems - The search engine asks for your feedback after visiting a site.
- Most feedback systems have severe flaws (but can be useful if implemented correctly and used with other methods)
- More sophisticated: Weighted Heuristic Page Analysis, Rank Merging, and Manipulation Prevention Systems

What Makes Ranking Optimization Hard?

- Link Spamming
- Keyword Spamming
- Page hijacking and URL redirection
- Intentionally inaccurate or misleading anchor text
- Accurately targeting people's expectations

PageRank - Section V - Outline

- Link Analysis and Anchors
- Introduction to PageRank
- Calculating Naive PR
- Example
- Calculating PR using Linear Algebra
- Implementing PageRank
- Problems with PR
- Improving PageRank

Link Analysis and Anchors

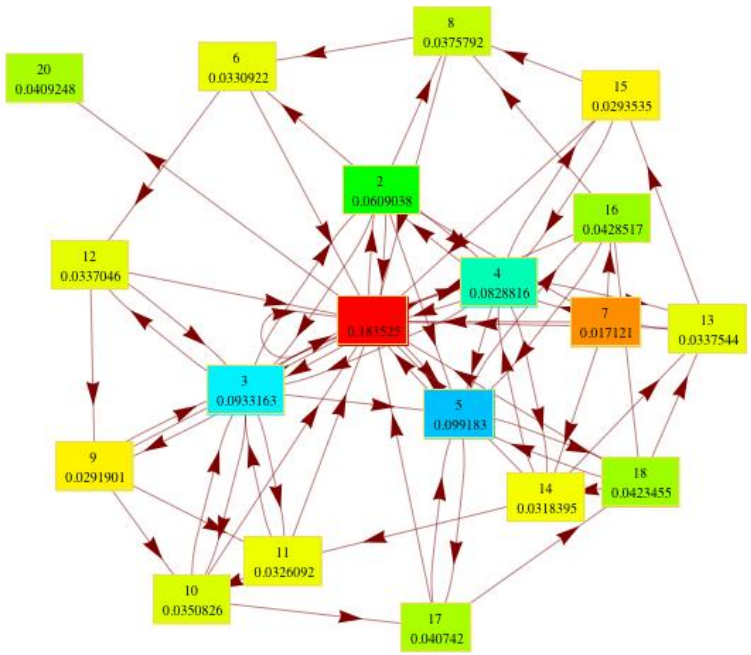
- Hypertextual Links are convenient to users and represent physical citations on the Web.
- Anchor Text Analysis:
  $< ahref = "http://www.google.com" >$Anchor Text$< /a >$
- Can be more accurate description of target site than target sites text itself
- Can point at non-HTTP or non-text; such as images, videos, databases, pdf's, ps's, etc.
- Also, anchors make it possible for non-crawled pages to be discovered.

Introduction to PageRank

- Rights belong to Google, patent belongs to Stanford University
- Top 10 IEEE ICDM data mining algorithm
- Algorithm used the rank the relative importance of pages within a network.
- PageRank idea based on the elements of democrating voting and citations.
- The PR Algorithm uses logarithmic scaling; the total PR of a network is 1.

Introduction to PageRank

- Rights belong to Google, patent belongs to Stanford University PageRank is a link analysis algorithm that ranks the relative importance of all web pages within a network. It does this by looking at three web page features:
  - 1. Outgoing Links - the number of links found in a page
  - 2. Incoming Links - the number of times other pages have sited this page
  - 3. Rank - A value representing the page's relative importance in the network



Calculating Naive PageRank

The Page Rank Equation

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)}) \quad (1)$$

$PR(A)$ = The PageRank of page A

C(A) or $L(A)$ = the total number of outgoing links from page A

d = The damping factor. Induces randomness to prevent certain pages from gaining too much rank. (1-d) ensures adds the values lost by multiplying by the damping factor to ensure the sum of all web pages in the network is 1. The damping factor also enforces a random surfing model which is comparable to Markov Chains.

Calculating Naive PageRank, Cont'd

The PageRank of a page A, denoted PR(A), is decided by the quality and quantity of sites linking or citing it. Every page T_i that links to page A is essentially casting a vote, deeming page A important. By doing this, T_i propogates some of it's PR to page A.

How can we determine how much rank rank an individual page T_i gives to A?

T_i may contain many links – not just a single link to page A.

T_i must propogate it's page rank equally to it's citations. Thus, we only want to give page A a fraction of the $PR(T_i)$.

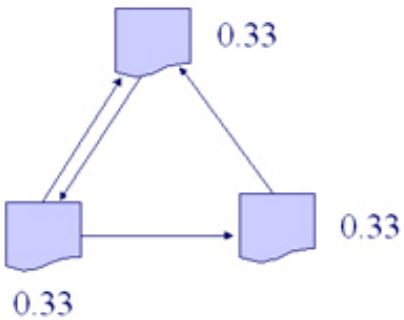
The amount of PR that T_i gives to A is be expressed as the damping value times the $PR(T_i)$ divided by the total number of outgoing links from T_i.

Naive Example

Computing PageRank (Step 0)

Initialize so total rank sums to 1.0

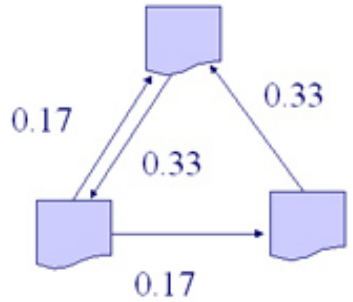
$$x_i^{(0)} = \frac{1}{n}$$



Computing PageRank (Step 1)

Propagate weights across out-edges

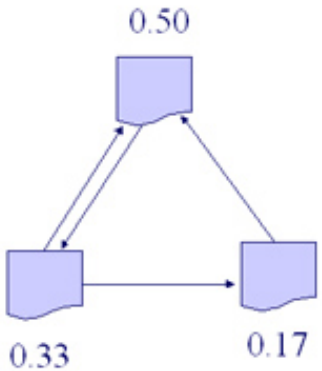
$$x_i^{(k+1)} = \sum_{j \in B_i} \frac{1}{N_j} x_j^{(k)}$$



Computing PageRank (Step 2)

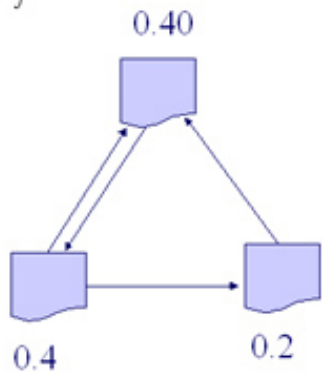
Compute weights based on in-edges

$$x_i^{(1)} = \sum_{j \in B_i} \frac{1}{N_j} x_j^{(0)}$$



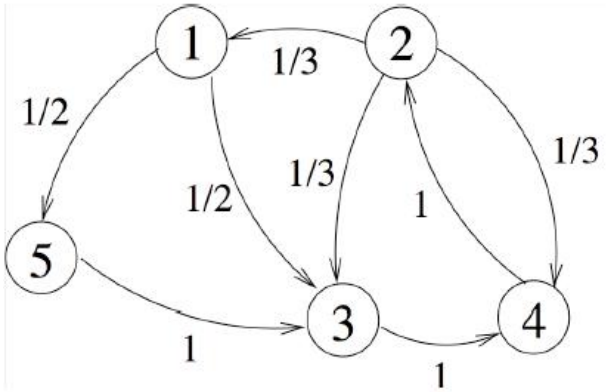
Computing PageRank (Convergence)

$$x_i^{(k+1)} = \sum_{j \in B_i} \frac{1}{N_j} x_j^{(k)}$$



www.seas.upenn.edu/~zives/cis555/slides/G-WebSearch.ppt

Calculating PageRank using Linear Algebra

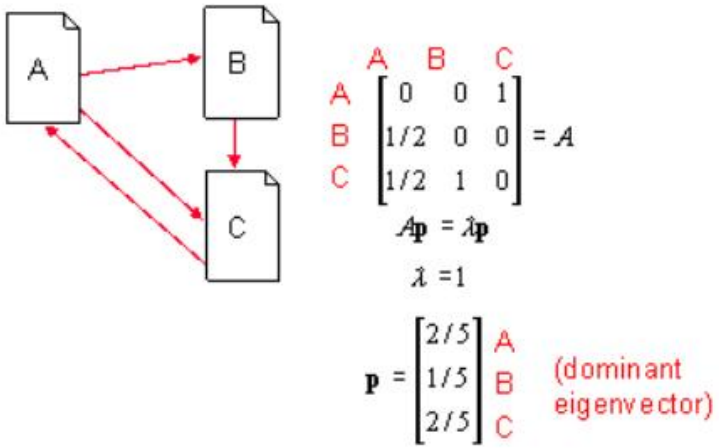


$$B = \begin{bmatrix} 0 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1/2 & 1/3 & 0 & 0 & 1 \\ 0 & 1/3 & 1 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Typically PageRank computation is done by finding the principal eigenvector of the Markov chain transition matrix. The vector is solved using the iterative power method. Above is a simple Naive PageRank setup which expresses the network as a link matrix.

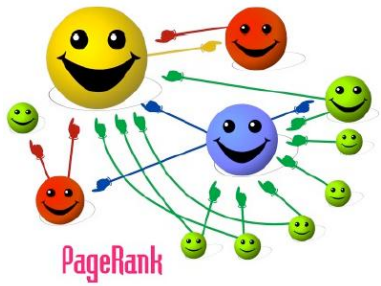
- More examples can be found at:
  - http://www.ianrogers.net/google-page-rank/
  - http://www.webworkshop.net/pagerank.html
  - http://www.math.uwaterloo.ca/ hdesterc/websiteW/ [...] Data/presentations/pres2008/ChileApr2008.pdf

Calculating PageRank using Linear Algebra, Cont'd



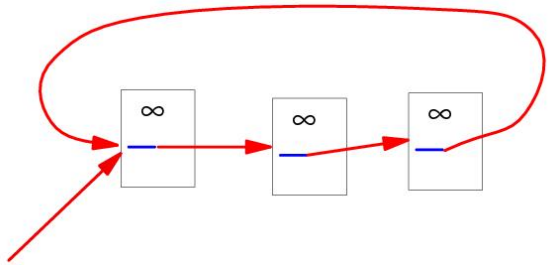
For those interested in the actual PageRank Calculation and Implementation process (involving heavier linear algebra), I invite you to view my "Additional Resources" slide.

Implementing PageRank



- Convert all page URLs into unique integers (for IDs)
- Sort link structure by Parent ID
- Dangling links removed from link db
- Several strategies: they choose iterate until convergence
- Assign initial values to nodes (based on assumed importance)
- Run the model until it converges. To ensure accuracy, repeat process with correct initialized weights.

Disadvantages and Problems



- Rank Sinks: Occur when pages get in infinite link cycles.
- Spider Traps: A group of pages is a spider trap if there are no links from within the group to outside the group.
- Dangling Links: A page contains a dangling link if the hypertext points to a page with no outgoing links.
- Dead Ends: are simply pages with no outgoing links.
- - Solution to all of the above: By introducing a damping factor, the figurative random surfer stops trying to traverse the sunk page(s) and will either follow a link randomly or teleport to a random node in the network.

Improving PageRank

- Based on, "The PageRank Citation Ranking: Bringing Order to the Web", choosing initial values for pages in a network will not affect the final values, just the rate of convergence.
- By choosing initial values wisely, the convergence of the algorithm will be quicker.
- Local Graph Partitioning using PageRank Vectors *complicated* http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04031383
- Distributed PR Comp. based on Iterative Aggregation-Dissagregation Methods (IAD).
- Alternate Infrastructure to avoid costly data-seeks (lookups)

Competition and Future Research - Section VI - Outline

- Jon Kleinberg's HITs
- Lempel and Moran's SALSA
- Ask.com's ExpertRank
- Microsoft's BrowserRank
- Yahoo!'s TrustRank (also from Stanford!)

Jon Kleinberg's HITs Algorithm

Hyperlink-Induced Topic Search (HITS) (also known as Hubs and authorities) is a link analysis algorithm that rates Web pages, developed by Jon Kleinberg. It determines two values for a page: its authority, which estimates the value of the content of the page, and its hub value, which estimates the value of its links to other pages.

It is processed on a small subset of .relevant. documents, not all documents as was the case with PageRank.

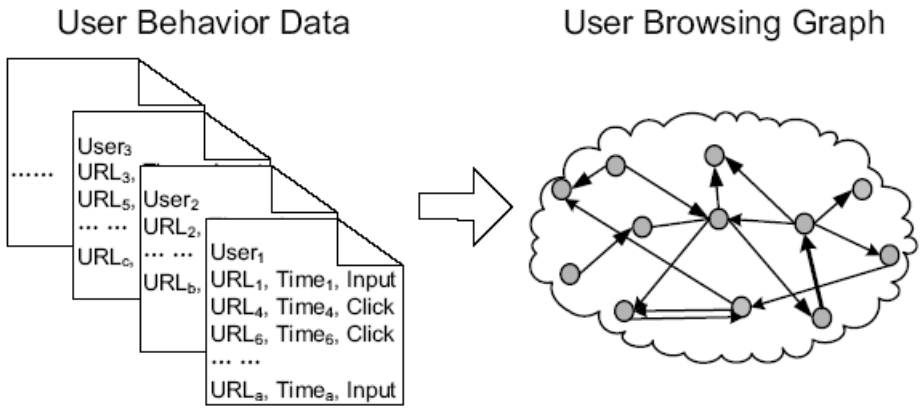
Teoma aka Ask.com's ExpertRank

Teomo was unique because of its link popularity algorithm. Unlike Google's PageRank, Teoma's technology (Subject-Specific Popularity) analyzed links in context to rank a web page's importance within its specific subject. For instance, a web page about 'baseball' would rank higher if other web pages about 'baseball' link to it.

Lampel and Moran's SALSA

SALSA, a new stochastic approach for link-structure analysis, which examines random walks on graphs derived from the link-structure. Read more at - http://portal.acm.org/citation.cfm?id=383041

Microsoft's BrowseRank



BrowseRank is based on the time a user spends on a site.

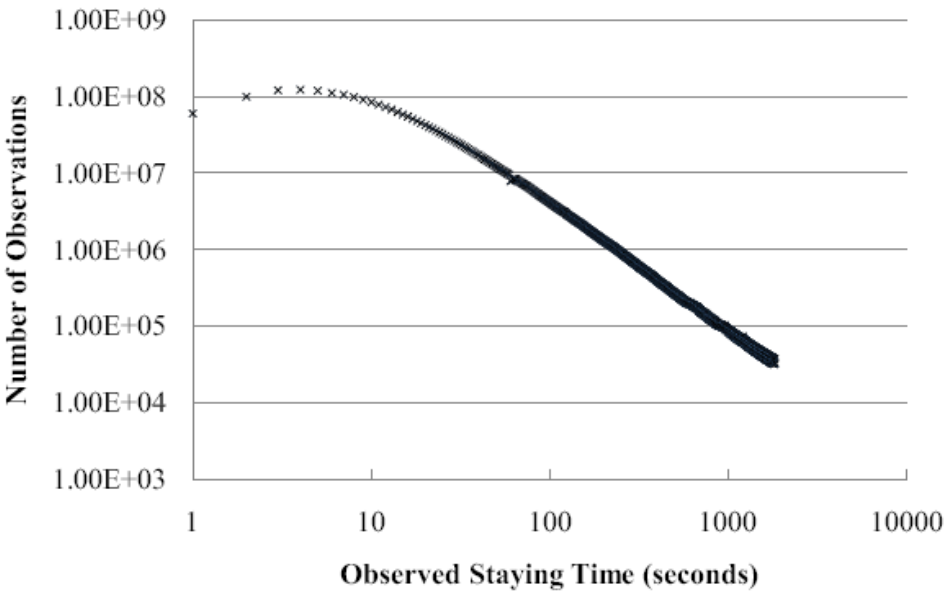


Table 3: Top 20 websites by three different algorithms

No.	PageRank	TrustRank	BrowseRank
1	adobe.com	adobe.com	*myspace.com*
2	passport.com	yahoo.com	msn.com
3	msn.com	google.com	yahoo.com
4	microsoft.com	msn.com	*youtube.com*
5	yahoo.com	microsoft.com	live.com
6	google.com	passport.net	*facebook.com*
7	mapquest.com	ufindus.com	google.com
8	miibeian.gov.cn	*sourceforge.net*	ebay.com
9	w3.org	*myspace.com*	*hi5.com*
10	godaddy.com	*wikipedia.org*	*bebo.com*
11	statcounter.com	phpbb.com	*orkut.com*

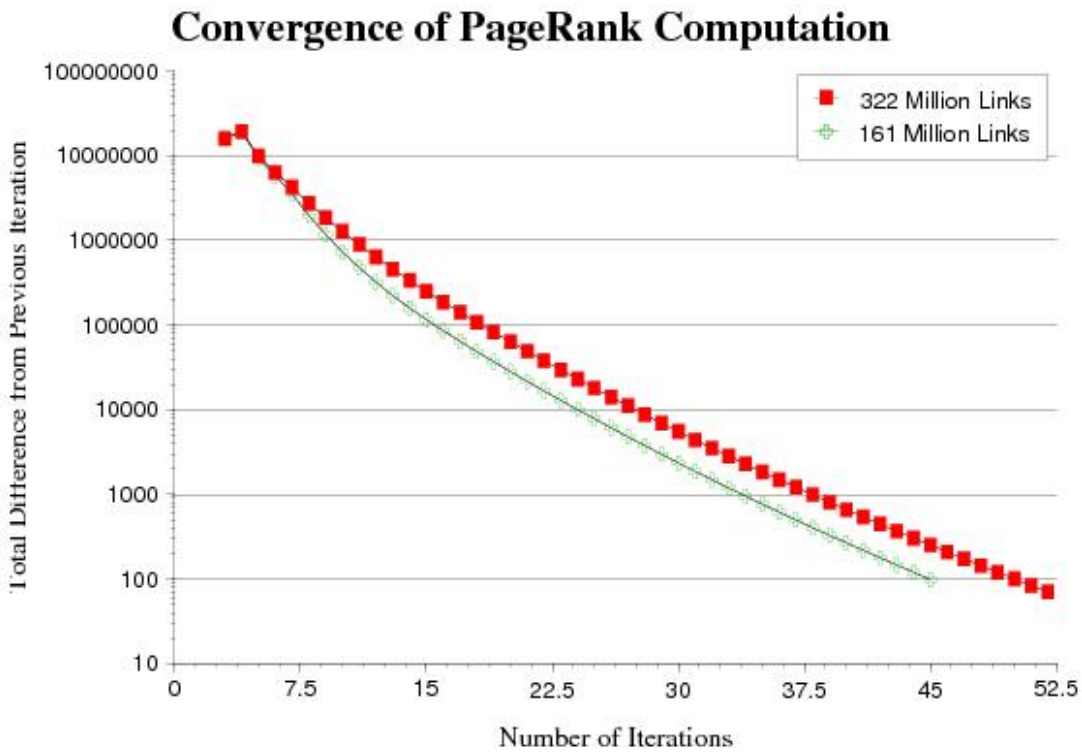
Yahoo!'s TrustRank (also from Stanford!)

TrustRank method calls for selecting a small set of seed pages to be evaluated by an expert. Once the reputable seed pages are manually identified, a crawl extending outward from the seed set seeks out similarly reliable and trustworthy pages. TrustRank's reliability diminishes as documents become further removed from the seed set.

Conclusion - Section VII - Outline

- Experimental Results (Benchmarking)
- The Future of PageRank and Other Applications
- Exam Questions
- Bibliography

Benchmarking Convergence



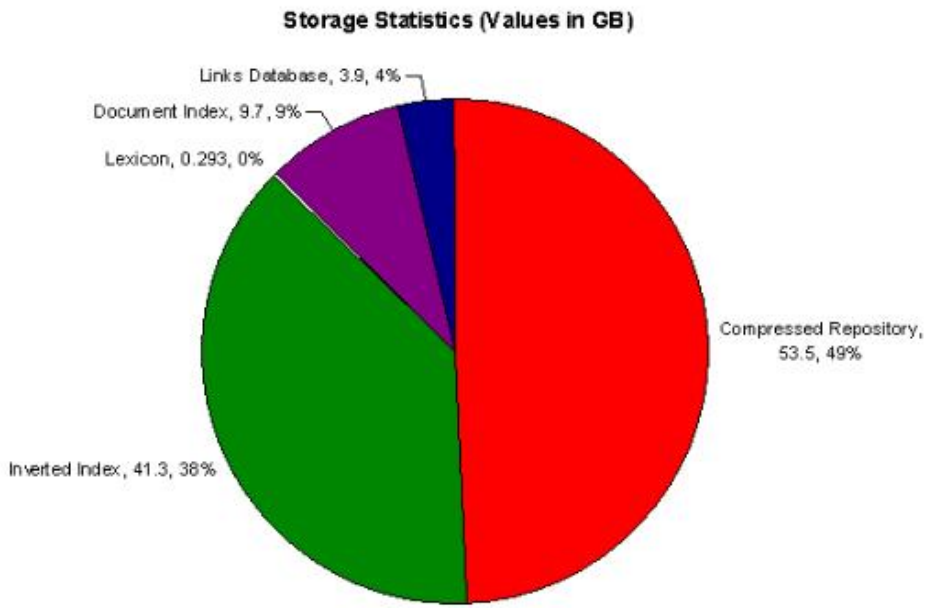
- convergence of the Power Method is FAST! 322 million links converge almost as quickly as 161 million.
- Doubling the size has very little effect on the convergence time.

Experimental Results

Storage Requirements

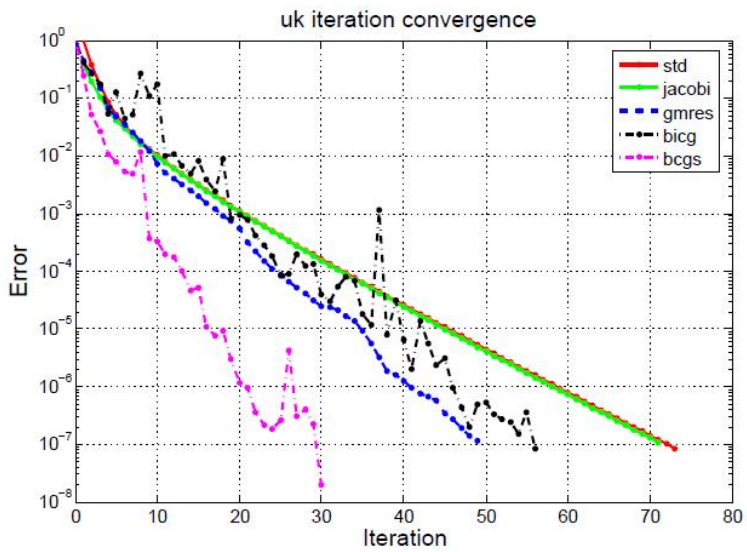
(from http://www.ics.uci.edu/~scott/google.htm)

At the time of publication, Google had the following statistical breakdown for storage requirements:

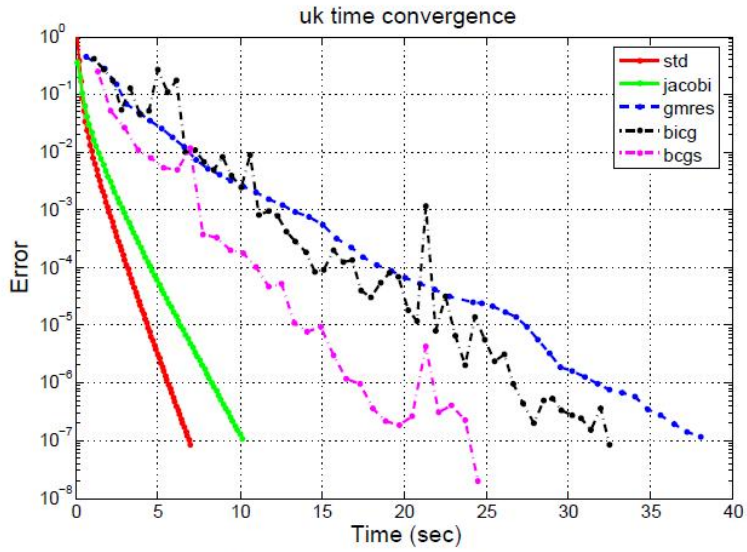


- Data structures obviously highly optimized for space
- Infrastructure setup for high parallelization.

Benchmarking of Advanced PageRank Implementations



(a) Convergence Iterations



(b) Convergence Time

- Parallel PageRank with PageRank Iterations versus Jacobi, GMRES, BiCG, and GiCGSTAB:
  http://www.stanford.edu/ dgleich/projects/ppagerank/index.html
- Compares convergence and time for different implementations:
  http://www.stanford.edu/ dgleich/publications/
  gleich-zhukov-sc-pprank-abstract.pdf
- http://www.cs.cmu.edu/ yangboz/cikm05_pagerank.ppt

Future of PR

- A version of PageRank has recently been proposed as a replacement for the traditional Institute for Scientific Information (ISI) impact factor,[13] and implemented at eigenfactor.org. Instead of merely counting total citation to a journal, the "importance" of each citation is determined in a PageRank fashion.

- Because PageRank roughly corresponds to a random web surver model, PR can be used to estimate web traffic.

Final Exam Questions

- (1) Please state the PageRank formula and describe it's components

The Page Rank Equation

$$PR(A) = (1-d) + d(\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)}) \quad (2)$$

$PR(A)$ = The PageRank of page A

C(A) or $L(A)$ = the total number of outgoing links from page A

d = The damping factor.

Final Exam Questions

- (2) Given a network of five node:rank pairs [a:.2,b:.3,c:.1,d:.3,e:1], explain how to generate the corresponding a Naive PR Link Matrix. Assume pages can link to themselves (there are not zeros along the diaganol)

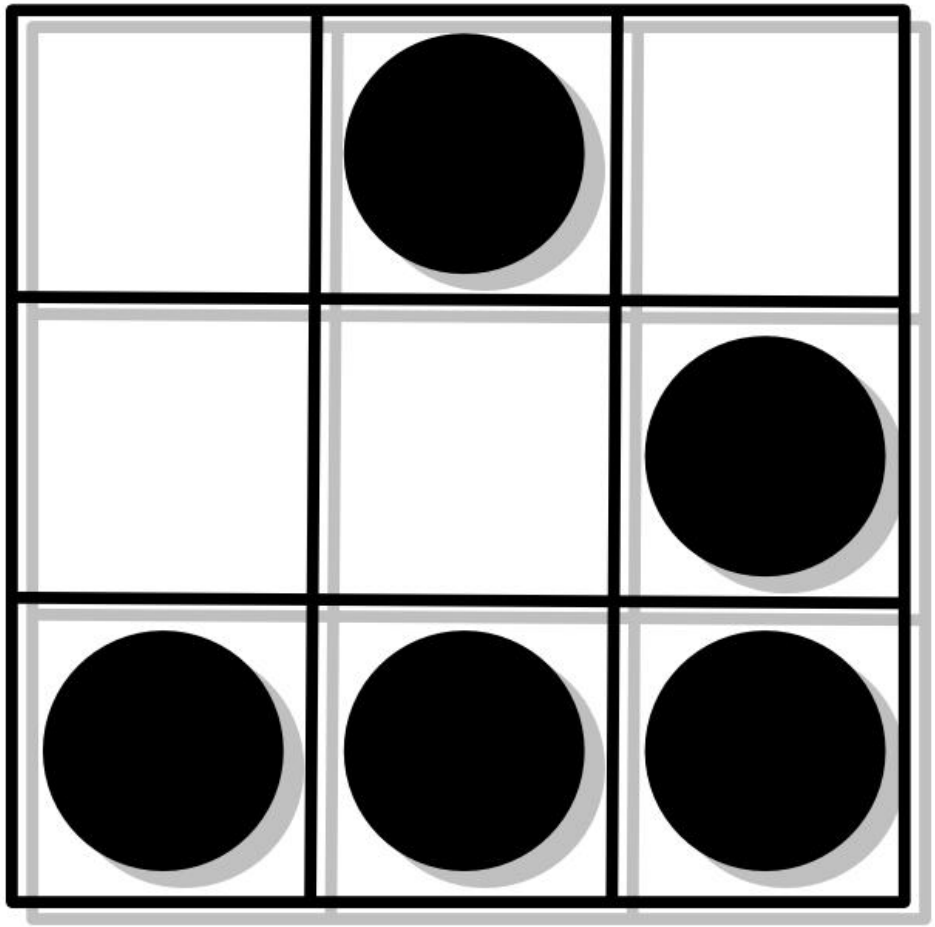
The answer will be a 5 by 5 matrix where the labels for the rows and columns are the pages in the network. The first element of every row belongs to the first page, ..., and the fifth and final element in every row belongs to the the fifth page. Likewise the first element in every column belongs to the first page and the fifth element of every column belongs to the fifth page. Each item in the matrix is thus correlated with two pages – it's row represents one page and it's column represents another. In this model, the value within any entry represents the fraction of rank the column page provides the row page. To fill in any particular matrix entry, find the node in the graph associated with the entry's column and count it's number of outgoing links. The entry's value is the column page's rank over this outgoing link count.

Final Exam Questions

- (3) Explain the signifiance of the damping factor. What values are it between? What problems does it solve? What factors does it account for?

The damping factor induces randomness to prevent certain pages from gaining too much rank. The value (1-d) from the PR equation ensures that the values lost by multiplying by the damping factor are added back to the system (such that the network maintains an accumulative PR of 1). The damping factor also enforces a random surfing model which is comparable to Markov Chains. The damping factor handles dead ends and infinit loops by offering a probability of a random teleport or link traversal.

Questions?



Additional Resources

- http://cis.poly.edu/suel/papers/pagerank.pdf - PR via The SplitAccumulate Algorithm, Merge-Sort, etc.
- http://nlp.stanford.edu/ manning/papers/PowerExtrapolation.pdf -PR via Power Extrapolation: includes benchmarking
- http://www.webworkshop.net/pagerank_calculator.php - neat little tool for PR calculation with a matrix
- http://www.miislita.com/information-retrieval-tutorial/ [...] matrix-tutorial-3-eigenvalues-eigenvectors.html

Bibliography

- http://www.math.uwaterloo.ca/ hdesterc/ websiteW/Data/presentations/pres2008/ChileApr2008.pdf
- Infrastructure Diagram and explanations from last year's slides
- Google Query Steps from last year's slides
- http://portal.acm.org/citation.cfm?id=1099705
- http://www.springerlink.com/content/ 60u6j88743wr5460/fulltext.pdf?page=1
- http://www.ianrogers.net/google-page-rank/
- http://www.seobook.com/microsoft-search- browserank-research-reviewed
- http://www.webworkshop.net/pagerank.html
- http://en.wikipedia.org/wiki/PageRank
- http://pr.efactory.de/e-pagerank-distribution.shtml
- http://www.cs.helsinki.fi/u/linden/teaching/irr06/ drafts/petteri_huuhka_google_draft.pdf
- http://www-db.stanford.edu/ backrub/pageranksub.ps